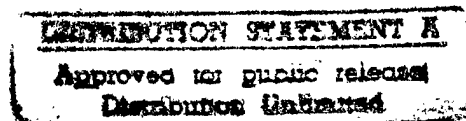


DATE: 4/01/97

CONTROLLING OFFICE FOR THIS DOCUMENT IS:
DIRECTOR, Army High Performance Computing
Research Center (AHPCRC)
Army Research Laboratory
Aberdeen, MD

POC: Director (Tayn E. Tezduyar)

DISTRIBUTION STATEMENT A: Public release



AHPCRC Researchers Demonstrate Heterogeneous Computing at Supercomputing '94

Wint 95

by Marek Behr (AHPCRC) and Alan Klietz (AHPCRC-MSCI)

Recent years have witnessed a vast increase in the number of computing platforms which are available to computational scientists. A multitude of design paths taken by the computer manufacturers in their never-ending quest for high performance and rapid programmability have resulted in a wide spectrum of CPU, memory and storage architectures. The choice of the optimal computational platform and algorithm for a given problem on the part of the numerical analyst has become much more important than in the past. While many applications significantly benefit from their adaptation to parallel distributed memory computers, some algorithms are inherently non-parallelizeable. Some algorithms, when moved from a vector supercomputer to a parallel architecture which they cannot fully exploit, suffer major degradation in performance. In this case the decision to continue the utilization of such algorithm on a traditional supercomputer may be the correct one.

Large, complex, simulations may require many different algorithms to perform the analysis, with sections of the code ranging from embarrassingly parallel to unavoidably sequential. It is here that the concept of heterogeneous computing may provide a welcome alternative to months needed to replace the sequential code, or to leaving the entire package, including the easily parallelizeable portions, on a non-parallel machine. In heterogeneous computation, many kinds of computers may cooperate, with each of the machines performing tasks for which it is particularly suited. The platforms exchange data utilizing a fast network connection.

One example of an application which may benefit from the heterogeneous computing approach is an implicit fluid flow calculation with a finite element method. One of the major tasks in this case is the construction of the (typically large) systems of equations which represent the fluid unknowns. With a correct selection of data structures and a proper initialization step, this task can proceed concurrently for all elements of the grid, exploiting the fine-grain parallelism inherent in the finite element approach. Another time-consuming step is the solution of the coupled equation systems. Iterative solution techniques, which offer parallelization potential, can only be applied in some of the cases. For some equation systems, e.g. those with an extremely high condition number, a direct solution method might be the only recourse. Unfortunately, implementation of such methods on distributed memory computers still poses a challenge, and using a vector machine for this step may be preferable. Therefore, a heterogeneous program might be decomposed as

19970401 107

follows: a) the formation of the equation system takes place on a massively parallel computer such as the CM-5, b) the components of the system are transmitted to a vector machine such as the Cray C90, where the system is assembled and solved, and c) the solution is transmitted back to CM-5 and used to update the field variables and begin the next iteration step.

There are numerous obstacles to seamless integration of such diverse platforms as the CM-5 and C90. High data transfer rate between the two machines is essential in order to avoid the introduction of another bottleneck obstacle to high performance. A High Performance Parallel Interface (HIPPI) network interface, available on most high performance computers, provides a bandwidth of 100 million bytes per second. This goes a long way towards alleviating the bottleneck. A HIPPI connection links the CM-5 and C90 operated by the Minnesota Supercomputer Center, and is used by the AHPARC researchers. Another problem stems from the fact that binary floating-point data formats used by different machines may be incompatible. In our example, the CM-5 employs the standard IEEE floating-point representation, the C90 uses a Cray-specific format, and the Cray T3D makes use of both IEEE and Cray formats. Finally, the methods of accessing the HIPPI interface are platform- and language-specific, causing the codes that attempt to use the native HIPPI libraries to become non-portable.

To simplify access to the heterogeneous computing engines, the second author developed the HIPPI/RPC library. The library is based on the Remote Procedure Call (RPC) concept. In this concept, a program running on a local machine performs a subroutine call that is executed on a remote machine. The actual arguments included in the subroutine call are transferred over a network connection to the remote machine, where they are accessed as dummy arguments by the subroutine. After the remote machine finishes execution of the subroutine, the modified values of the dummy arguments are copied back to replace the arguments in the local calling program. The specification of the remote procedure can be expressed using standard Fortran 77 or Fortran 90 notation. The HIPPI/RPC library is designed to be easy to use, allowing the rapid prototyping of new applications.

All data format conversion and translation, including floating-point format conversion, is performed by the system. HIPPI/RPC can be used on top of a variety of communication protocols and hardware. HIPPI is preferred because of its high performance, but TCP/IP over Ethernet or FDDI is also possible. The transport selection can be performed at runtime.

In order to use the HIPPI/RPC interface, the following steps are required: a) decomposition of the program into subroutines, some of which are executed on one or more remote machines, b) creation of a HIPPI/RPC Language (HRPCL) source file to declare the arguments of the remote subroutines, c) compilation of the HRPCL source file using the hrpcgen protocol compiler, d) compilation

of the hrpcgen-generated C and Fortran source files, along with user files, into the master program on the local machine and the server program on the remote machine. After starting the server program, the master program can be run.

The aforementioned example of the finite element fluid flow computation was used by AHPCRC researchers Tayfun Tezduyar, Shahrouz Aliabadi and the first author, in collaboration with ARL researchers Gloria Wren and Steve Ray, to demonstrate the HIPPI/RPC capabilities at the Supercomputing 94 conference, which took place in Washington, D.C. on November 15-17, 1994. The simulation involved the flow inside a regenerative liquid propellant gun (RLPG). The master program, a compressible flow finite element code based on a stabilized space-time formulation, ran on CM-5 and was coupled with a C90 implementation of a skyline direct solver. The schematics of the RLPG, and the Mach number field inside the combustion chamber at one instant during the computation, is shown in Figure 1. Because of their dependence on the direct solver, the RLPG simulations are typically performed on the C90. The Supercomputing 94 demonstration showed the possibility of moving the majority of the code to the potentially faster CM-5, while keeping the modules which do not benefit from parallelism on the Cray C90.

To learn more about HIPPI/RPC, please refer to the "HIPPI/RPC User's Guide", AHPCRC publication UG-0008 9/91 available from the AHPCRC by calling (612) 626-1550 or sending an email, with a request, staff@ahpcrc.umn.edu